BlueFinder

Final Presentation

DEVELOPMENT TEAM

Stefan Crigler

PCB design, software help, team lead

Robert Tremewan

PCB design, software help

Renny Hong

software design, RF/DSP

Arthur Lobins

software design, RF/DSP

Cynthia Alvarez

software design, Fusion/UI

BlueFinder is a prototype hardware/software platform that enables limited-range direction tracking of Bluetooth devices without requiring any additional information.

OVERVIEW

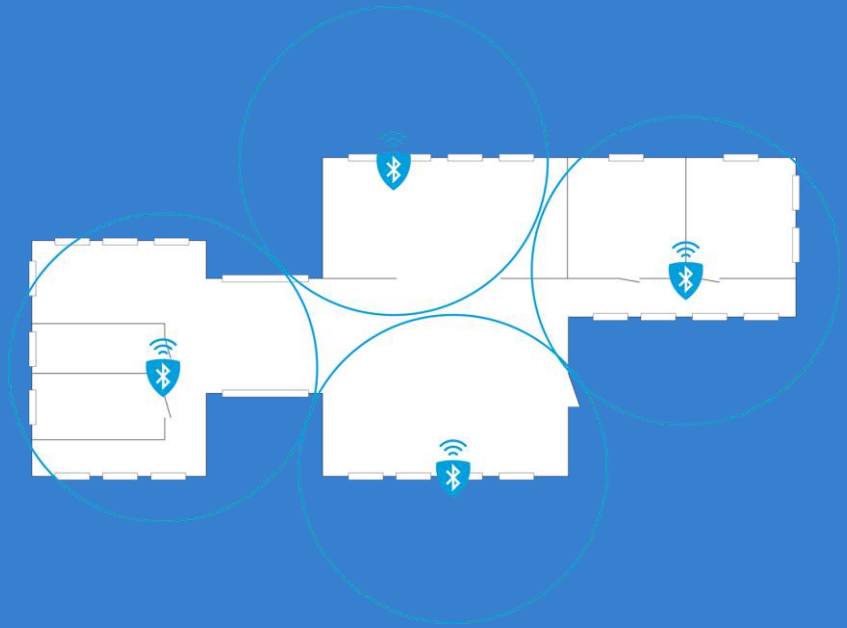
- Building on last year's capstone project BlueDentist, which captured Bluetooth packet information
- Locate Bluetooth devices using XTRX software defined radios (SDR) programmed with direction-finding algorithms

POTENTIAL APPLICATIONS

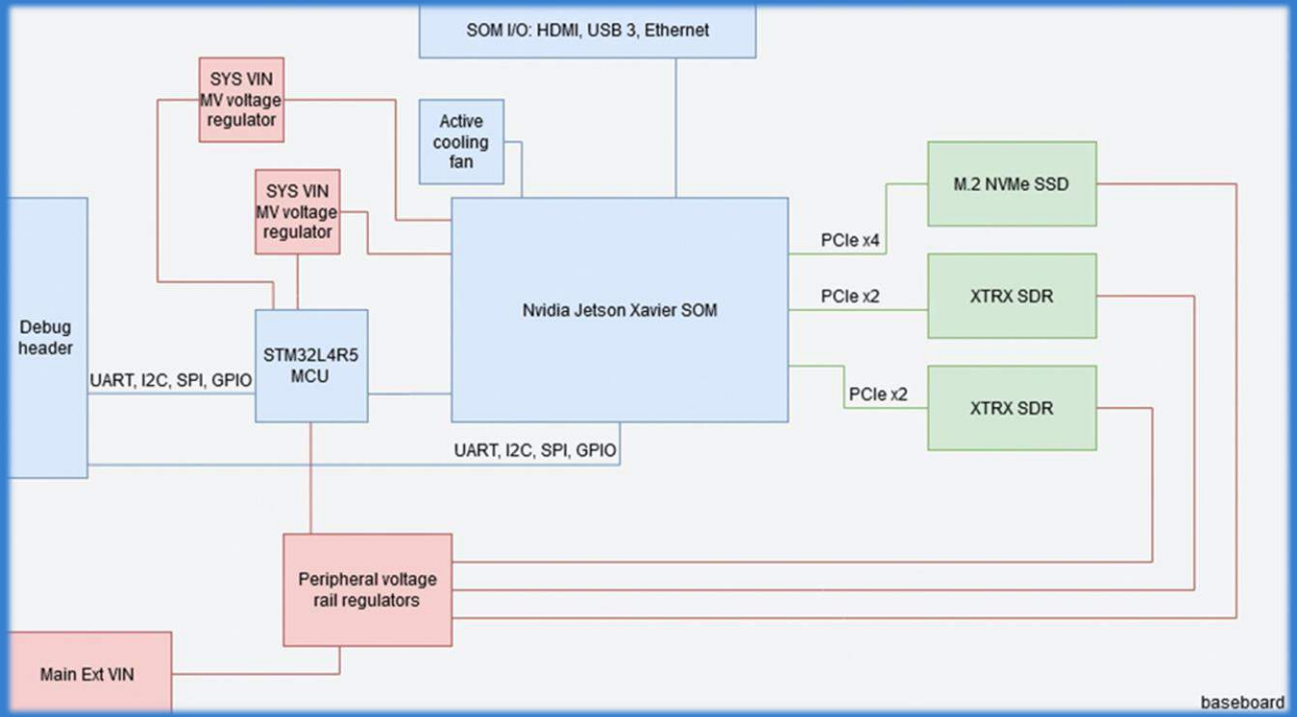
POTENTIAL APPLICATIONS

Tracking and analysis of ad-hoc gatherings of customers in a restricted location

Tracking, tallying, and management of workers or assets in a workplace



BLOCK DIAGRAM



- STM32 controls power
- XTRX SDR send raw direction data to Jetson

KEY HARDWARE

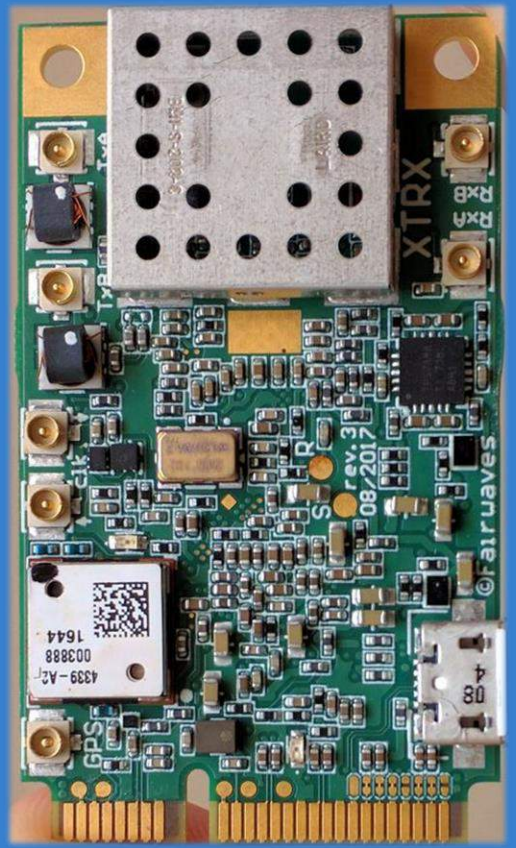
Nvidia Jetson AGX Xavier

- Runs the software which processes the raw data coming in from the radios
- Software was developed on the devboard version while the custom PCB was developed

KEY HARDWARE

XTRX

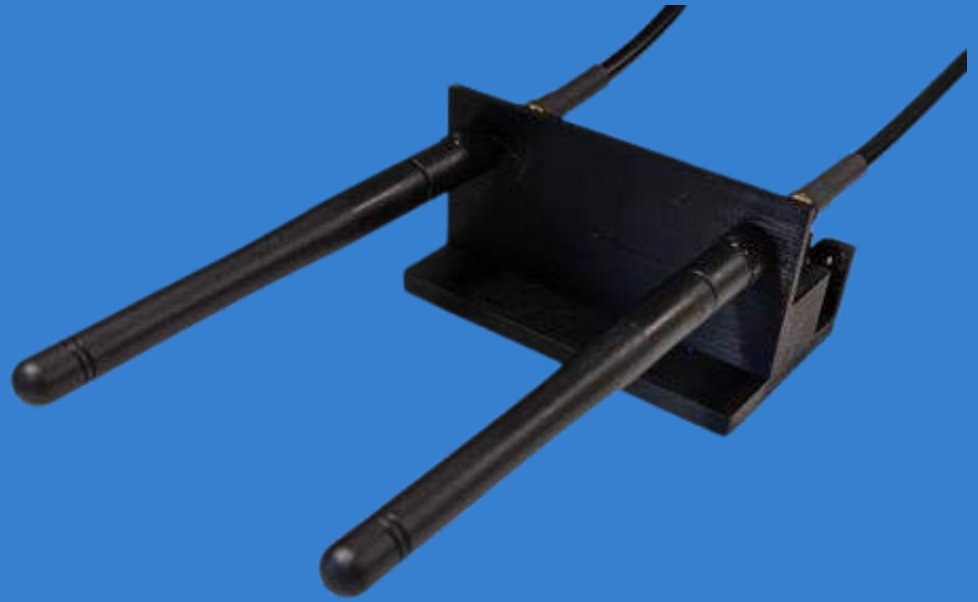
- Software-defined radio programmed to listen on the Bluetooth frequencies around 2.4GHz
- We have two on the board for up to four potential antennae, however only one was used in our current setup



KEY HARDWARE

Antenna Assembly

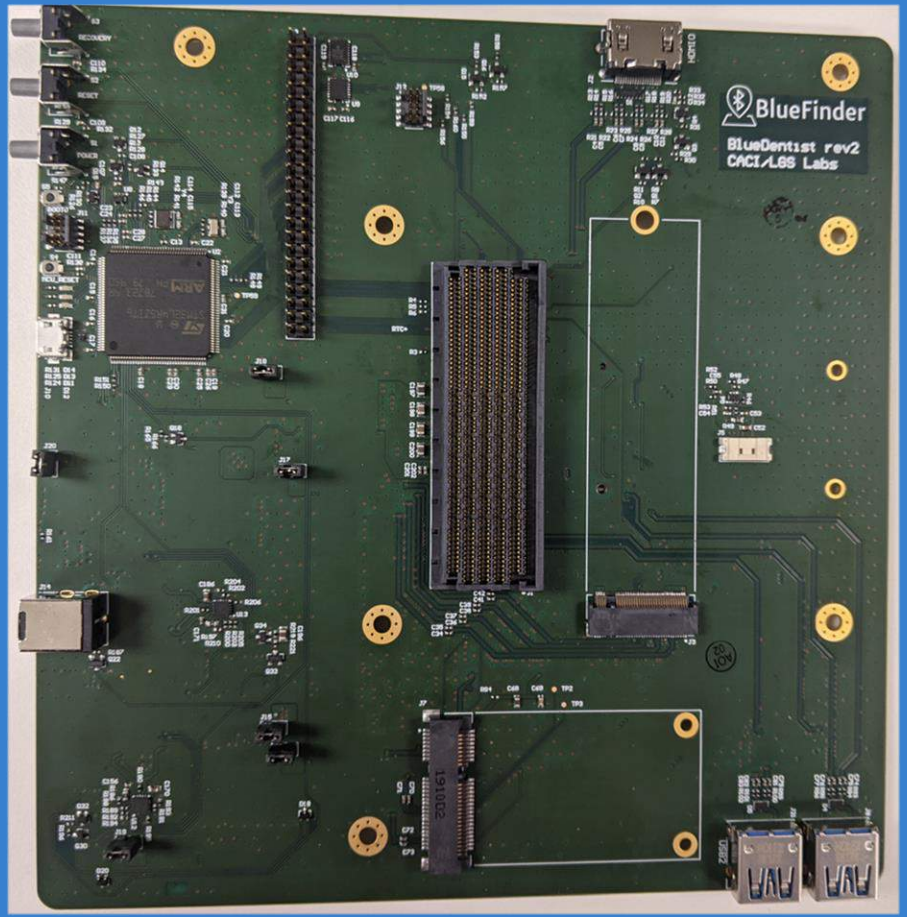
- In order to position the antennas exactly as required we had to design and 3D print a set of mounts
- 6.5 cm apart, fixable positions for consistent antennae positioning



KEY HARDWARE

Revision 2 PCB

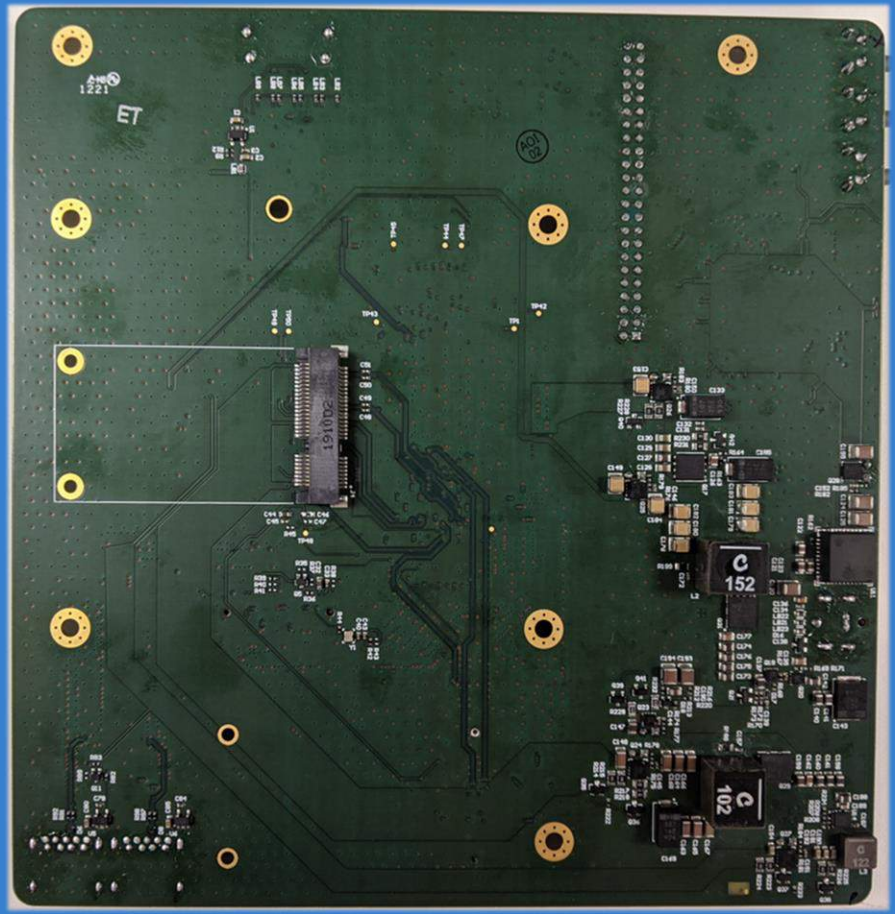
- We needed an extra PCIe slot to accommodate both SDRs so we had to revise the board designed by the BlueDentist project last year



KEY HARDWARE

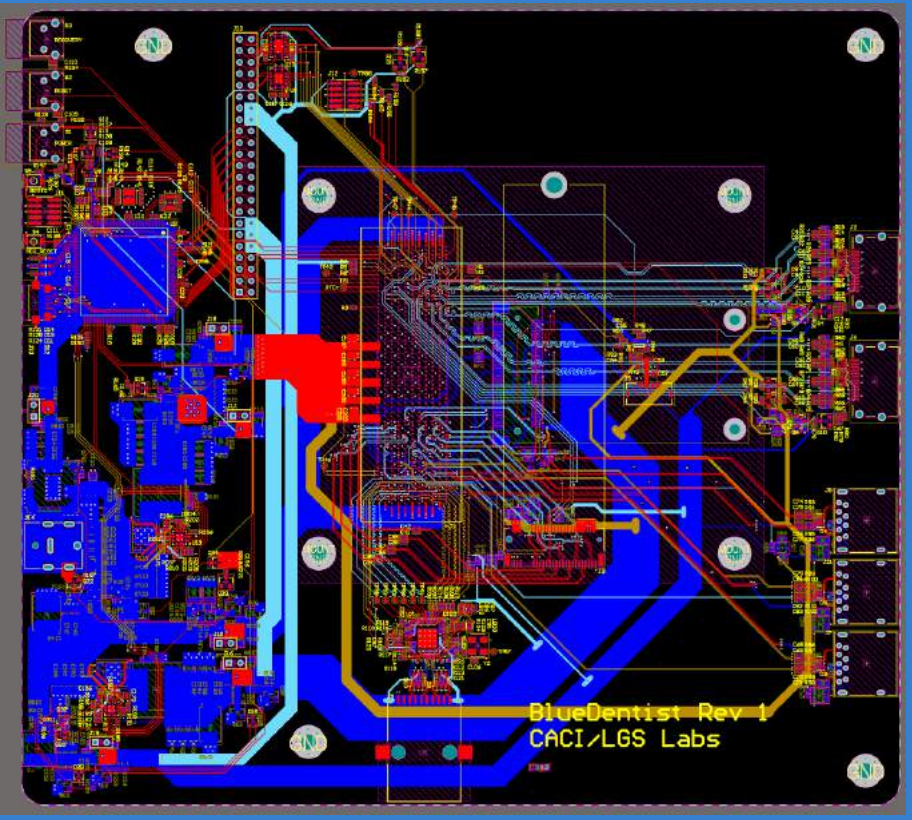
Revision 2 PCB

- We needed an extra PCIe slot to accommodate both SDRs so we had to revise the board designed by the BlueDentist project last year

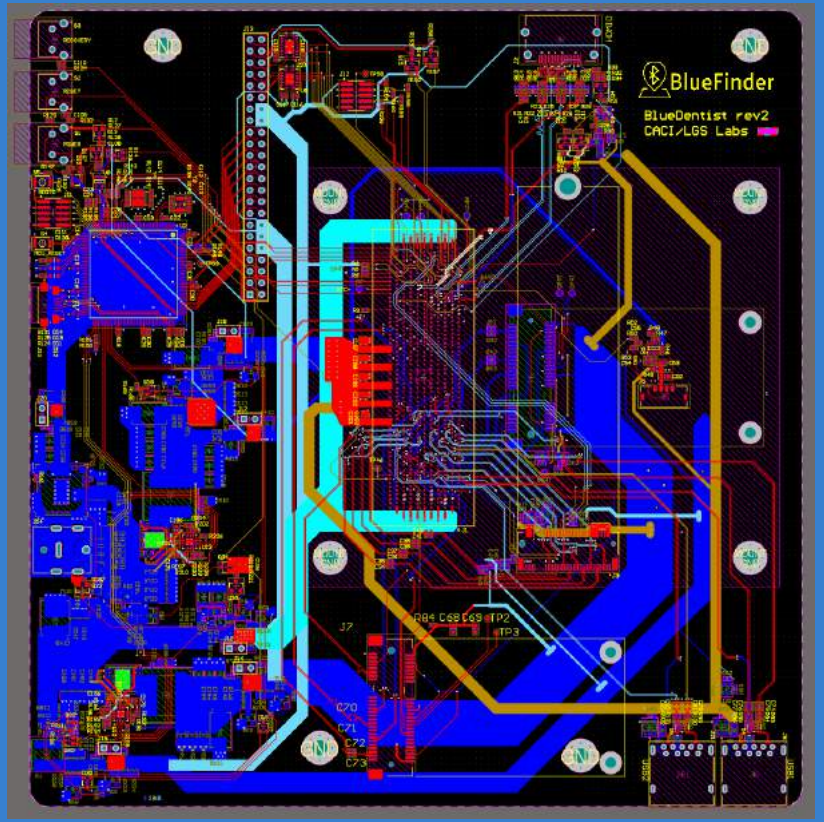


Revisions compared

OLD



NEW

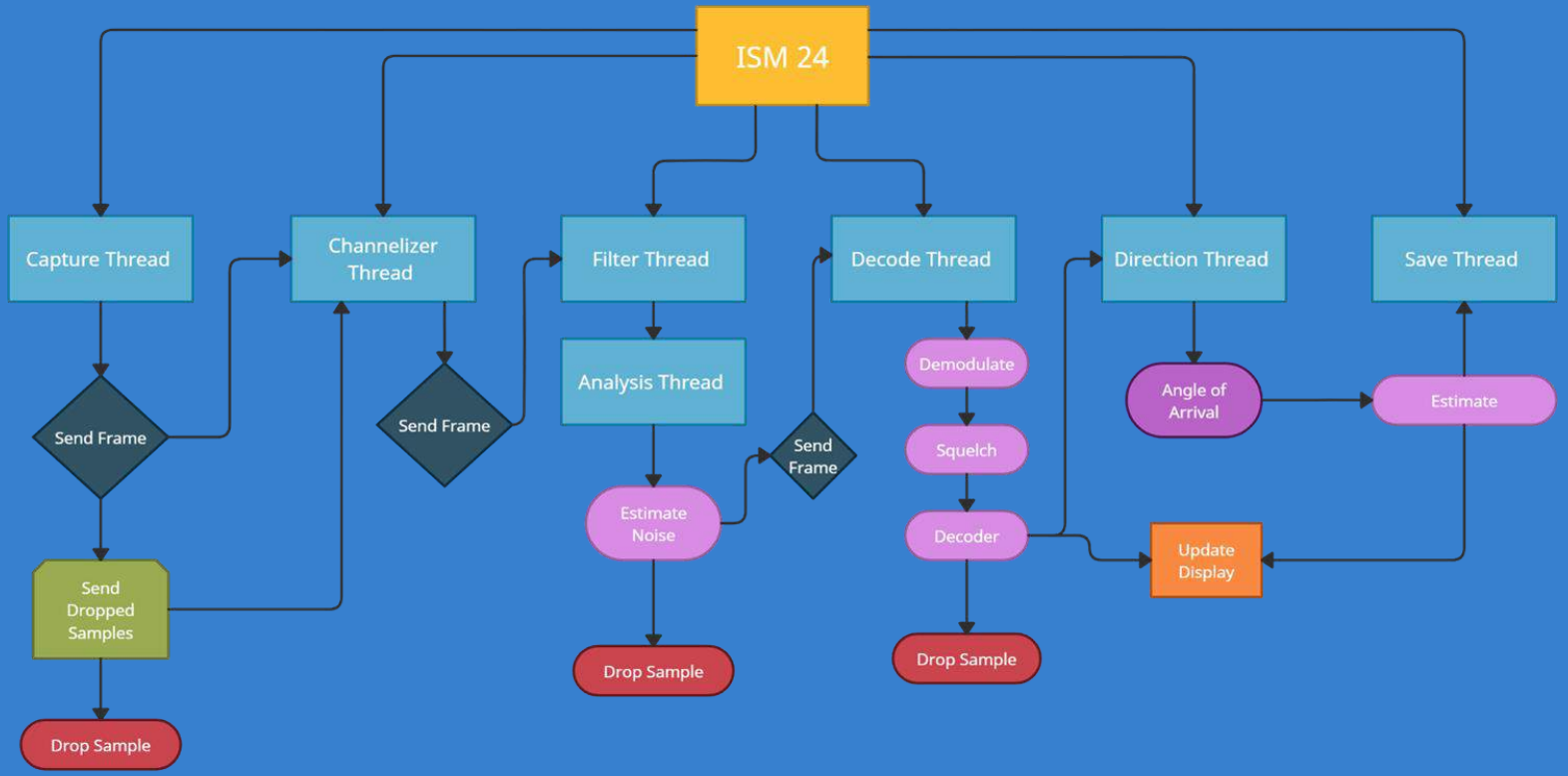


Revisions compared

OLD

NEW

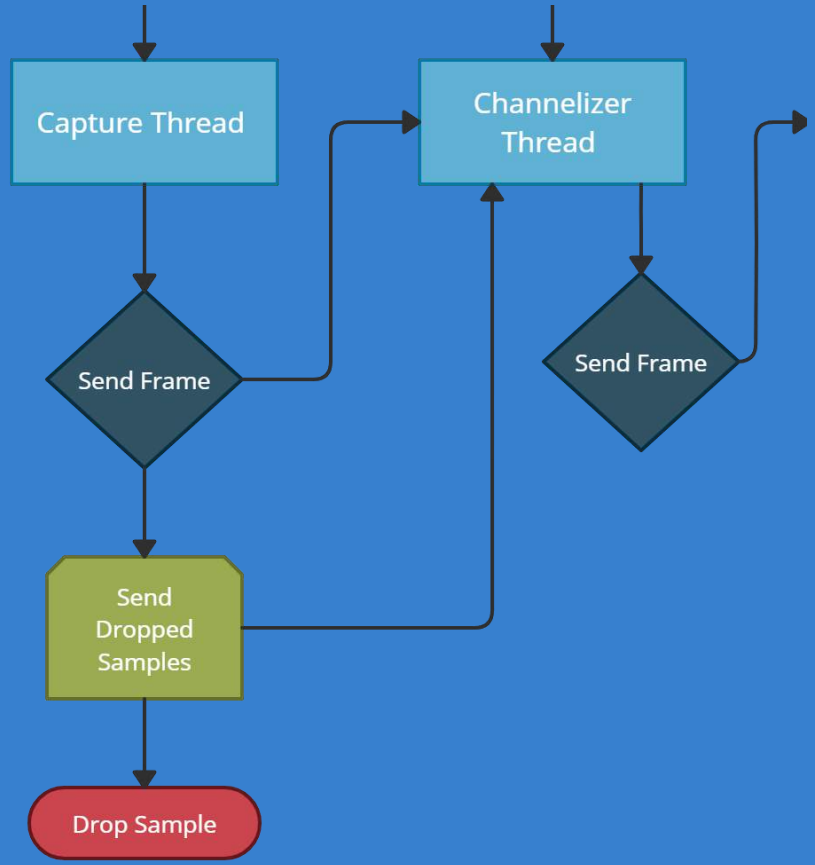
SOFTWARE FLOW DIAGRAM



SOFTWARE FLOW DIAGRAM

Capture and Channelizer Threads

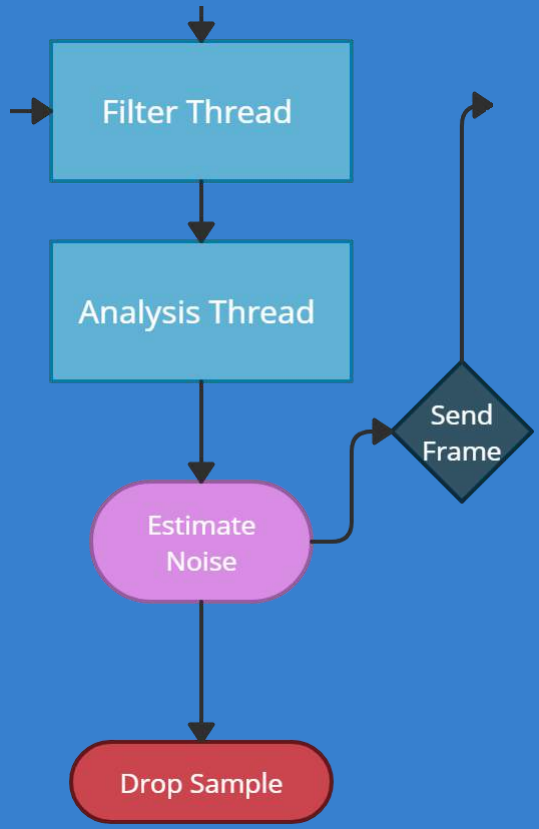
- The Capture Thread is where we adjust the radio configuration such as sample size and frequency
- The captured data is placed into a frame, which acts as a buffer for complex 16-bit numbers. If the buffer is full, some samples will be dropped, or sent along with the next frame
- A captured frame is then sent on to the channelizer thread, which splits the sample up into the 79 bluetooth channels



SOFTWARE FLOW DIAGRAM

Filter and Analysis Threads

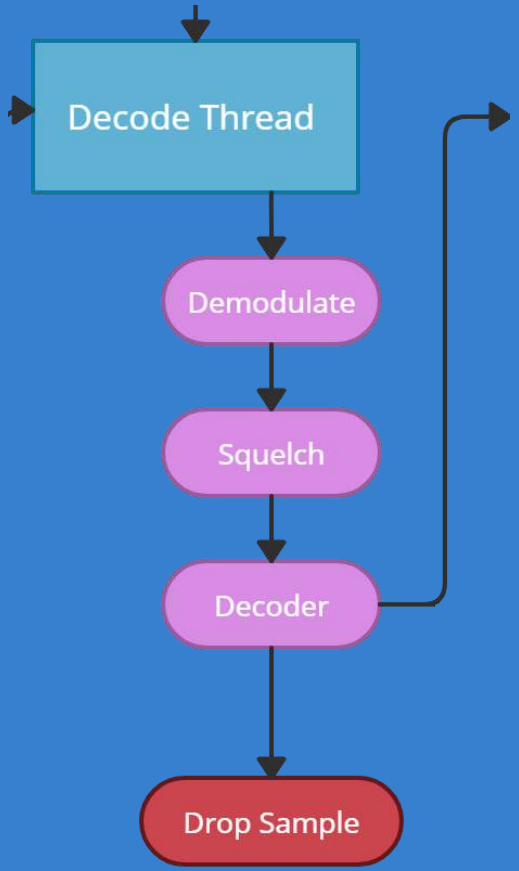
- The Filter thread determines the time selection of the frame that contains a Bluetooth signal
- The Analysis thread will calculate the channel power for the frame and estimate noise floor threshold
- If the power of the channel > noise threshold, then it is possible a Bluetooth device is in a channel



SOFTWARE FLOW DIAGRAM

Decode thread

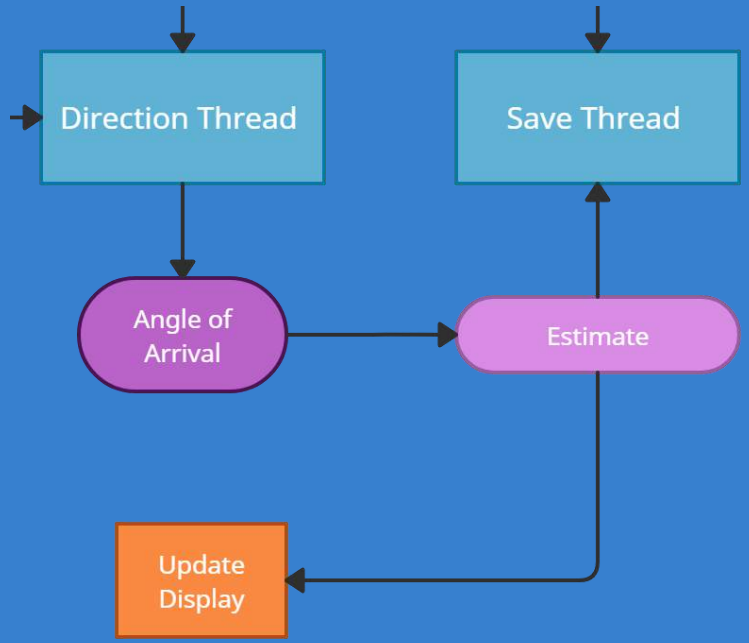
- If a frame potentially captured a Bluetooth signal, then it is sent to the decode thread, where the frame is decoded for a Bluetooth access code
- The access code is estimated based on the number of bit flips required in the capture frame to form a valid Bluetooth access code



SOFTWARE FLOW DIAGRAM

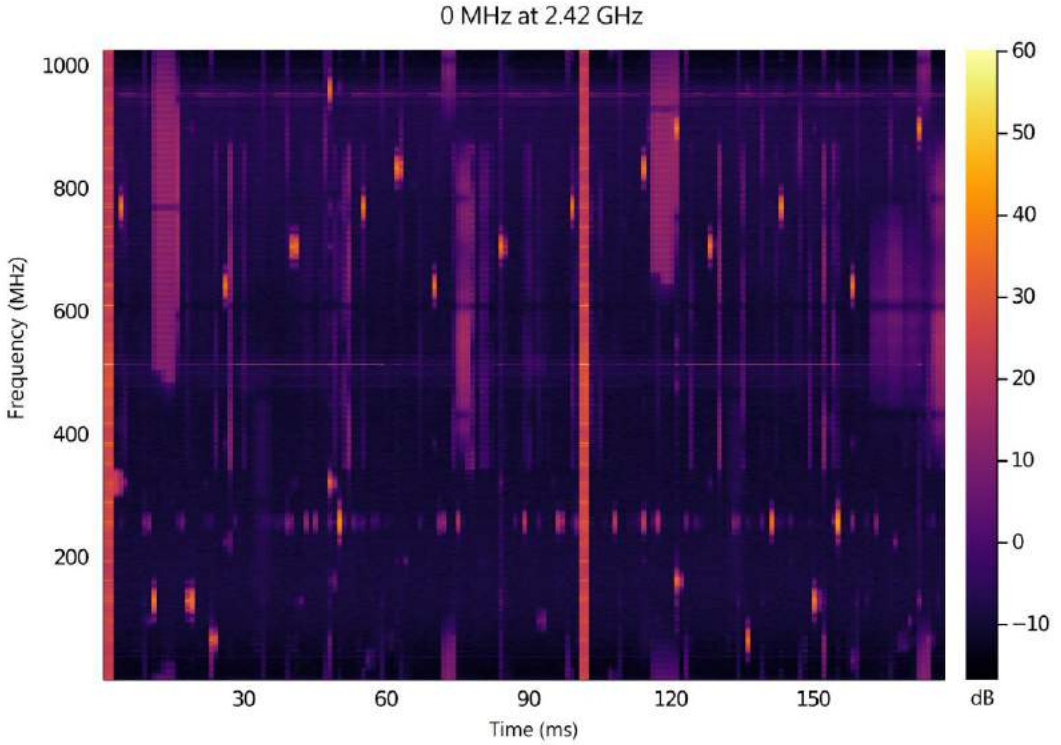
Direction and Save Threads

- If a frame is found to contain a Bluetooth access code, the frame is forwarded to the Direction Thread
- Calculates AoA estimation using MUSIC algorithm on the channelized frame
- Finally, data is saved locally on an SSD.



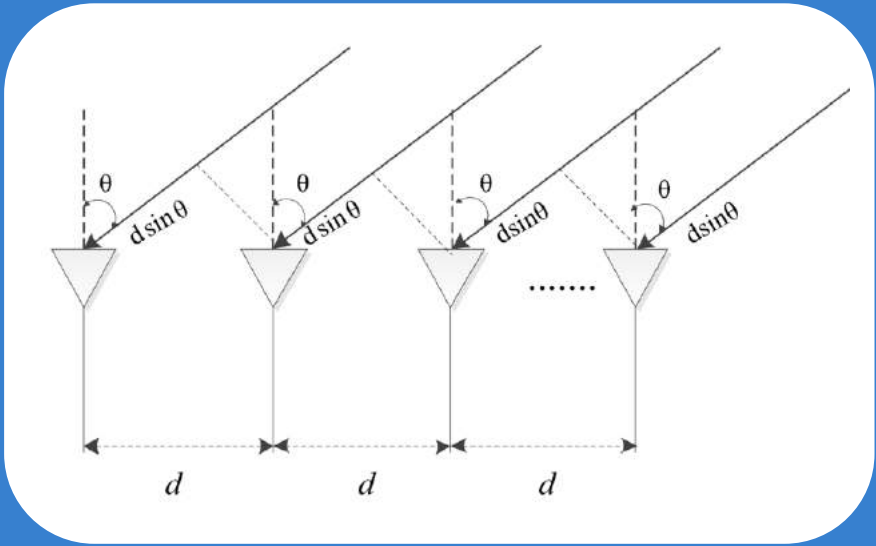
DATA COLLECTION

- Monitoring the 2.4GHz band for interesting bursts of signals
- The data is processed if it contains a Bluetooth signal under a certain noise threshold
- Interesting frames are decoded and used in our direction finding algorithm



ANGLE OF ARRIVAL ALGORITHM

- Linear array of antennas with known spacing
- The bearing angle to the signal source is determined using phase difference between the signals received by multiple antennas
- MUtiple SIgnal Classification Algorithm (MUSIC)
  - High Resolution
  - Possible to estimate AoA for multiple signals simultaneously



MUSIC ALGORITHM

- Estimate the autocorrelation matrix using an eigenspace method

1. Calculate sample covariance matrix
2. Eigendecomposition

    - Largest eigenvalues and corresponding eigenvectors span the signal subspace
    - The rest corresponds to noise space

- Signal vectors in the signal subspace must be orthogonal to the noise space

3. Generate complex sinusoids of various incident angles
4. Measure the level of orthogonality w.r.t. noise space

- Angle resulting in the highest orthogonality is the estimated direction of the signal source

UC **SANTA BARBARA**

College of Engineering

Thank you for your time!
Questions?